

Über die Klassifizierung von Knoten in dynamischen Netzwerken mit Inhalt

Martin Thoma

Zusammenfassung—In dieser Arbeit wird der DYCOS-Algorithmus, wie er in [AL11] vorgestellt wurde, erklärt. Er arbeitet auf Graphen, deren Knoten teilweise mit Beschriftungen versehen sind und ergänzt automatisch Beschriftungen für Knoten, die bisher noch keine Beschriftung haben. Dieser Vorgang wird „Klassifizierung“ genannt. Dazu verwendet er die Struktur des Graphen sowie textuelle Informationen, die den Knoten zugeordnet sind. Die in [AL11] beschriebene experimentelle Analyse ergab, dass er auch auf dynamischen Graphen mit 19 396 bzw. 806 635 Knoten, von denen nur 14 814 bzw. 18 999 beschriftet waren, innerhalb von weniger als einer Minute auf einem Kern einer Intel Xeon 2.5 GHz CPU mit 32 G RAM ausgeführt werden kann. Zusätzlich wird [AL11] kritisch Erörtert und es werden mögliche Erweiterungen des DYCOS-Algorithmus vorgeschlagen.

Keywords: DYCOS, Label Propagation, Knotenklassifizierung

I. EINLEITUNG

Im Folgenden werden in Abschnitt I-A einige Beispiele, in denen der DYCOS-Algorithmus Anwendung finden könnte, dargelegt. In Abschnitt I-B wird die Problemstellung formal definiert und in Abschnitt I-C wird auf besondere Herausforderungen der Aufgabenstellung hingewiesen.

A. Motivation

Teilweise beschriftete Graphen sind allgegenwärtig. Publikationsdatenbanken mit Publikationen als Knoten, Literaturverweisen und Zitaten als Kanten sowie von Nutzern vergebene Beschriftungen (sog. *Tags*) oder Kategorien als Knotenbeschriftungen; Wikipedia mit Artikeln als Knoten, Links als Kanten und Kategorien als Knotenbeschriftungen sowie soziale Netzwerke mit Eigenschaften der Benutzer als Knotenbeschriftungen sind drei Beispiele dafür. Häufig sind Knotenbeschriftungen nur teilweise vorhanden und es ist wünschenswert die fehlenden Knotenbeschriftungen automatisiert zu ergänzen.

B. Problemstellung

Gegeben ist ein Graph, dessen Knoten teilweise beschriftet sind. Zusätzlich stehen zu einer Teilmenge der Knoten Texte bereit. Gesucht sind nun Knotenbeschriftungen für alle Knoten, die bisher noch nicht beschriftet sind.

Definition 1 (Knotenklassifizierungsproblem) Sei

$G_t = (V_t, E_t, V_{L,t})$ ein gerichteter Graph, wobei V_t die Menge aller Knoten, $E_t \subseteq V_t \times V_t$ die Kantenmenge und $V_{L,t} \subseteq V_t$ die Menge der beschrifteten Knoten jeweils zum Zeitpunkt t bezeichne. Außerdem sei L_t die Menge aller zum Zeitpunkt t vergebenen

Knotenbeschriftungen und $f : V_{L,t} \rightarrow L_t$ die Funktion, die einen Knoten auf seine Beschriftung abbildet.

Weiter sei für jeden Knoten $v \in V$ eine (eventuell leere) Textmenge $T(v)$ gegeben.

Gesucht sind nun Beschriftungen für $V_t \setminus V_{L,t}$, also $\tilde{f} : V_t \setminus V_{L,t} \rightarrow L_t$. Die Aufgabe, zu G_t die Funktion \tilde{f} zu finden heißt Knotenklassifizierungsproblem.

C. Herausforderungen

Die Graphen, für die dieser Algorithmus konzipiert wurde, sind viele 10 000 Knoten groß und dynamisch. „Dynamisch“ bedeutet in diesem Kontext, dass neue Knoten und eventuell auch neue Kanten hinzu kommen bzw. Kanten oder Knoten werden entfernt werden. Außerdem stehen textuelle Inhalte zu den Knoten bereit, die bei der Klassifikation genutzt werden können. Bei kleinen Änderungen sollte nicht alles nochmals berechnen werden müssen, sondern basierend auf zuvor berechneten Knotenbeschriftungen sollte die Klassifizierung angepasst werden.

II. RELATED WORK

Sowohl das Problem der Knotenklassifikation, als auch das der Textklassifikation, wurde bereits in verschiedenen Kontexten analysiert. Jedoch scheinen bisher entweder nur die Struktur des zugrundeliegenden Graphen oder nur Eigenschaften der Texte verwendet worden zu sein.

So werden in [BCM11], [SJ01] unter anderem Verfahren zur Knotenklassifikation beschrieben, die wie der in [AL11] vorgestellte DYCOS-Algorithmus, um den es in dieser Ausarbeitung geht, auch auf Random Walks basieren.

Obwohl es auch zur Textklassifikation einige Paper gibt [ZG02], [JCSZ10], geht doch keines davon auf den Spezialfall der Textklassifikation mit einem zugrundeliegenden Graphen ein.

Die vorgestellten Methoden zur Textklassifikation variieren außerdem sehr stark. Es gibt Verfahren, die auf dem bag-of-words-Modell basieren [Ko12] wie es auch im DYCOS-Algorithmus verwendet wird. Aber es gibt auch Verfahren, die auf dem Expectation-Maximization-Algorithmus basieren [NMTM99] oder Support Vector Machines nutzen [Joa98].

Es wäre also gut Vorstellbar, die Art und Weise wie die Texte in die Klassifikation des DYCOS-Algorithmus einfließen zu variieren. Allerdings ist dabei darauf hinzuweisen, dass die im Folgenden vorgestellte Verwendung der Texte sowohl einfach zu implementieren ist und nur lineare Vorverarbeitungszeit in Anzahl der Wörter des Textes hat, als auch es erlaubt einzelne Knoten zu klassifizieren, wobei der Graph nur lokal um den zu klassifizierenden Knoten betrachten werden muss.

III. DYCOS

A. Überblick

DYCOS (DYnamic Classification algorithm with cOntent and Structure) ist ein Knotenklassifizierungsalgorithmus, der ursprünglich in [AL11] vorgestellt wurde.

Ein zentrales Element des DYCOS-Algorithmus ist der sog. *Random Walk*:

Definition 2 (Random Walk, Sprung) Sei $G = (V, E)$ mit $E \subseteq V \times V$ ein Graph und $v_0 \in V$ ein Knoten des Graphen.

Ein Random Walk der Länge l auf G , startend bei v_0 ist nun der zeitdiskrete stochastische Prozess, der v_i auf einen zufällig gewählten Nachbarn v_{i+1} abbildet (für $i \in 0, \dots, l-1$). Die Abbildung $v_i \mapsto v_{i+1}$ heißt ein Sprung.

Der DYCOS-Algorithmus klassifiziert einzelne Knoten, indem r Random Walks der Länge l , startend bei dem zu klassifizierenden Knoten v gemacht werden. Dabei werden die Beschriftungen der besuchten Knoten gezählt. Die Beschriftung, die am häufigsten vorgekommen ist, wird als Beschriftung für v gewählt. DYCOS nutzt also die sog. Homophilie, d. h. die Eigenschaft, dass Knoten, die nur wenige Hops von einander entfernt sind, häufig auch ähnlich sind [BCM11]. Der DYCOS-Algorithmus arbeitet jedoch nicht direkt auf dem Graphen, sondern erweitert ihn mit Hilfe der zur Verfügung stehenden Texte. Wie diese Erweiterung erstellt wird, wird im Folgenden erklärt.

Für diese Erweiterung wird zuerst das Vokabular W_t bestimmt, das charakteristisch für eine Knotengruppe ist. Wie das gemacht werden kann und warum nicht einfach jedes Wort in das Vokabular aufgenommen wird, wird in Abschnitt III-D erläutert. Nach der Bestimmung des Vokabulars wird für jedes Wort im Vokabular ein Wortknoten zum Graphen hinzugefügt. Alle Knoten, die der Graph zuvor hatte, werden nun „Strukturknoten“ genannt. Ein Strukturknoten v wird genau dann mit einem Wortknoten $w \in W_t$ verbunden, wenn w in einem Text von v vorkommt. Abbildung 1 zeigt beispielhaft den so entstehenden, semi-bipartiten Graphen. Der DYCOS-Algorithmus betrachtet also die Texte, die einem Knoten zugeordnet sind, als eine Multimenge von Wörtern. Das heißt, zum einen wird nicht auf die Reihenfolge der Wörter geachtet, zum anderen wird bei Texten eines Knotens nicht zwischen verschiedenen Texten unterschieden. Jedoch wird die Anzahl der Vorkommen jedes Wortes berücksichtigt.

Entsprechend werden zwei unterschiedliche Sprungtypen unterschieden, die strukturellen Sprünge und inhaltliche Zweifachsprünge:

Definition 3 (struktureller Sprung) Sei $G_{E,t} = (V_t, E_{S,t} \cup E_{W,t}, V_{L,t}, W_t)$ der um die Wortknoten W_t erweiterte Graph.

Dann heißt das zufällige wechseln des aktuell betrachteten Knoten $v \in V_t$ zu einem benachbarten Knoten $w \in V_t$ ein struktureller Sprung.

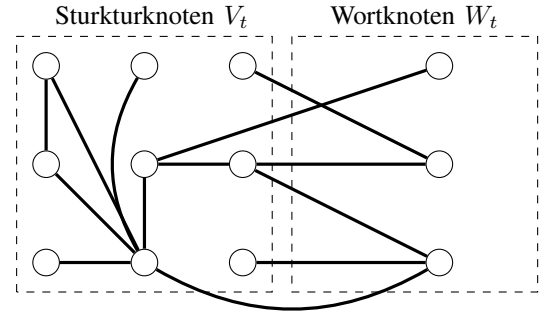


Abbildung 1: Erweiterter Graph

Im Gegensatz dazu benutzen inhaltliche Zweifachsprünge tatsächlich die Grapherweiterung:

Definition 4 (inhaltlicher Zweifachsprung) Sei

$G_t = (V_t, E_{S,t} \cup E_{W,t}, V_{L,t}, W_t)$ der um die Wortknoten W_t erweiterte Graph.

Dann heißt das zufällige wechseln des aktuell betrachteten Knoten $v \in V_t$ zu einem benachbarten Knoten $w \in W_t$ und weiter zu einem zufälligem Nachbar $v' \in V_t$ von w ein inhaltlicher Zweifachsprung.

Jeder inhaltliche Zweifachsprung beginnt und endet also in einem Strukturknoten, springt über einen Wortknoten und ist ein Pfad der Länge 2.

Ob in einem Sprung der Random Walks ein struktureller Sprung oder ein inhaltlicher Zweifachsprung gemacht wird, wird jedes mal zufällig neu entschieden. Dafür wird der Parameter $0 \leq p_S \leq 1$ für den Algorithmus gewählt. Mit einer Wahrscheinlichkeit von p_S wird ein struktureller Sprung durchgeführt und mit einer Wahrscheinlichkeit von $(1 - p_S)$ ein modifizierter inhaltlicher Zweifachsprung, wie er in Abschnitt III-C erklärt wird, gemacht. Der Parameter p_S gibt an, wie wichtig die Struktur des Graphen im Verhältnis zu den textuellen Inhalten ist. Bei $p_S = 0$ werden ausschließlich die Texte betrachtet, bei $p_S = 1$ ausschließlich die Struktur des Graphen.

Die Vokabularbestimmung kann zu jedem Zeitpunkt t durchgeführt werden, muss es aber nicht.

In Algorithmus 1 steht der DYCOS-Algorithmus in Form von Pseudocode: In Zeile 8 wird für jeden unbeschrifteten Knoten durch die folgenden Zeilen eine Beschriftung gewählt.

Zeile 10 führt r Random Walks durch. In Zeile 11 wird eine temporäre Variable für den aktuell betrachteten Knoten angelegt.

In Zeile 12 bis Zeile 21 werden einzelne Random Walks der Länge l durchgeführt, wobei die beobachteten Beschriftungen gezählt werden und mit einer Wahrscheinlichkeit von p_S ein struktureller Sprung durchgeführt wird.

B. Datenstrukturen

Zusätzlich zu dem gerichteten Graphen $G_t = (V_t, E_t, V_{L,t})$ verwaltet der DYCOS-Algorithmus zwei weitere Datenstruktu-

Algorithmus 1 DYCOS-Algorithmus

Input:

- 1: $G_{E,t} = (V_t, E_{S,t} \cup E_{W,t}, V_{L,t}, W_t)$ (Erweiterter Graph),
- 2: r (Anzahl der Random Walks),
- 3: l (Länge eines Random Walks),
- 4: p_s (Wahrscheinlichkeit eines strukturellen Sprungs),
- 5: q (Anzahl der betrachteten Knoten in der Clusteranalyse)

Output: Klassifikation von $V_t \setminus V_{L,t}$

```

6:
7:
8: for each Knoten  $v \in V_t \setminus V_{L,t}$  do
9:    $d \leftarrow$  leeres assoziatives Array
10:  for  $i = 1, \dots, r$  do
11:     $w \leftarrow v$ 
12:    for  $j = 1, \dots, l$  do
13:       $sprungTyp \leftarrow \text{RANDOM}(0, 1)$ 
14:      if  $sprungTyp \leq p_s$  then
15:         $w \leftarrow \text{STURKTURELLERSPRUNG}(w)$ 
16:      else
17:         $w \leftarrow \text{INHALTLICHERZWEIFACH-}$ 
18:         $\text{SPRUNG}(w)$ 
19:         $beschriftung \leftarrow w.\text{GETLABEL}()$ 
20:        if  $!d.\text{HASKEY}(beschriftung)$  then
21:           $d[beschriftung] \leftarrow 0$ 
22:           $d[beschriftung] \leftarrow d[beschriftung] + 1$ 
23:        if  $d.\text{ISEMPTY}()$  then  $\triangleright$  Es wurde kein beschrifteter
24:        Knoten gesehen
25:           $M_H \leftarrow \text{HÄUFIGSTELABELIMGRAPH}()$ 
26:        else
27:           $M_H \leftarrow \text{MAX}(d)$ 
28:         $\triangleright$  Wähle aus der Menge der häufigsten Beschriftungen
29:         $M_H$  zufällig eine aus
30:         $label \leftarrow \text{RANDOM}(M_H)$ 
31:         $v.\text{ADDLABEL}(label)$   $\triangleright$  und weise dieses  $v$  zu
32: return Beschriftungen für  $V_t \setminus V_{L,t}$ 

```

ren:

- Für jeden Knoten $v \in V_t$ werden die vorkommenden Wörter, die auch im Vokabular W_t sind, und deren Anzahl gespeichert. Das könnte z. B. über ein assoziatives Array (auch „dictionary“ oder „map“ genannt) geschehen. Wörter, die nicht in Texten von v vorkommen, sind nicht im Array. Für alle vorkommenden Wörter ist der gespeicherte Wert zum Schlüssel $w \in W_t$ die Anzahl der Vorkommen von w in den Texten von v .
- Für jedes Wort des Vokabulars W_t wird eine Liste von Knoten verwaltet, in deren Texten das Wort vorkommt. Diese Liste wird bei den inhaltlichen Zweifachsprung, der in Abschnitt III-C erklärt wird, verwendet.

C. Sprungtypen

Die beiden bereits definierten Sprungtypen, der strukturelle Sprung sowie der inhaltliche Zweifachsprung werden im Folgenden erklärt.

Der strukturelle Sprung entspricht einer zufälligen Wahl eines Nachbarknotens, wie es in Algorithmus 2 gezeigt wird.

Algorithmus 2 Struktureller Sprung

```

1: function STURKTURELLERSPRUNG(Knoten  $v$ , Anzahl  $q$ )
2:    $n \leftarrow v.\text{NEIGHBORCOUNT}$   $\triangleright$  Wähle aus der Liste der
3:   Nachbarknoten
4:    $r \leftarrow \text{RANDOMINT}(0, n - 1)$   $\triangleright$  einen zufällig aus
5:    $v \leftarrow v.\text{NEXT}(r)$   $\triangleright$  Gehe zu diesem Knoten
6:   return  $v$ 

```

Bei inhaltlichen Zweifachsprüngen ist jedoch nicht sinnvoll so strikt nach der Definition vorzugehen, also direkt von einem strukturellem Knoten $v \in V_t$ zu einem mit v verbundenen Wortknoten $w \in W_t$ zu springen und von diesem wieder zu einem verbundenem strukturellem Knoten $v' \in V_t$. Würde dies gemacht werden, wäre zu befürchten, dass aufgrund von Homonymen die Qualität der Klassifizierung verringert wird. So hat „Brücke“ im Deutschen viele Bedeutungen. Gemeint sein können z. B. das Bauwerk, das Entwurfsmuster der objektorientierten Programmierung oder ein Teil des Gehirns.

Deshalb wird für jeden Knoten v , von dem aus ein inhaltlicher Zweifachsprung gemacht werden soll folgende Textanalyse durchgeführt:

C1 Gehe alle in v startenden Random Walks der Länge 2 durch und erstelle eine Liste L der erreichbaren Knoten v' . Speichere außerdem, durch wie viele Pfade diese Knoten v' jeweils erreichbar sind.

C2 Betrachte im Folgenden nur die Top- q Knoten bzgl. der Anzahl der Pfade von v nach v' , wobei $q \in \mathbb{N}$ eine zu wählende Konstante des DYCOS-Algorithmus ist.¹ Diese Knotenmenge heiße im Folgenden $T(v)$ und $p(v, v')$ sei die Anzahl der Pfade von v über einen Wortknoten nach v' .

C3 Wähle mit Wahrscheinlichkeit $\frac{p(v, v')}{\sum_{w \in T(v)} p(v, w)}$ den Knoten $v' \in T(v)$ als Ziel des Zweifachsprungs.

Konkret könnte also ein inhaltlicher Zweifachsprung sowie wie in Algorithmus 3 beschrieben umgesetzt werden. Der Algorithmus bekommt einen Startknoten $v \in V_T$ und einen $q \in \mathbb{N}$ als Parameter. q ist ein Parameter der für den DYCOS-Algorithmus zu wählen ist. Dieser Parameter beschränkt die Anzahl der möglichen Zielknoten $v' \in V_T$ auf diejenigen q Knoten, die v bzgl. der Textanalyse am ähnlichsten sind.

In Zeile 2 bis Zeile 7 wird Punkt C1 durchgeführt und alle erreichbaren Knoten in *reachableNodes* mit der Anzahl der Pfade, durch die sie erreicht werden können, gespeichert.

¹Sowohl für den DBLP, als auch für den CORA-Datensatz wurde in [AL11, S. 364] $q = 10$ gewählt.

In Zeile 8 wird Punkt C2 durchgeführt. Ab hier gilt

$$|T| = \begin{cases} q & \text{falls } |\text{reachableNodes}| \geq q \\ |\text{reachableNodes}| & \text{sonst} \end{cases}$$

Bei der Wahl der Datenstruktur von T ist zu beachten, dass man in Zeile 23 über Indizes auf Elemente aus T zugreifen können muss.

In Zeile 10 bis 15 wird ein assoziatives Array erstellt, das von $v' \in T(v)$ auf die relative Häufigkeit bzgl. aller Pfade von v zu Knoten aus den Top- q abbildet.

In allen folgenden Zeilen wird Punkt C3 durchgeführt. In Zeile 17 bis Zeile 24 wird ein Knoten $v' \in T(v)$ mit einer Wahrscheinlichkeit, die seiner relativen Häufigkeit am Anteil der Pfade der Länge 2 von v nach v' über einen beliebigen Wortknoten entspricht ausgewählt und schließlich zurückgegeben.

Algorithmus 3 Inhaltlicher Zweifachsprung

```

1: function INHALTLICHERZWEIFACHSPRUNG(Knoten  $v \in V_T$ ,  $q \in \mathbb{N}$ )
2:    $\text{erreichbareKnoten} \leftarrow$  leeres assoziatives Array
3:   for each Wortknoten  $w$  in  $v.\text{GETWORDNODES}()$  do
4:     for each Strukturknoten  $x$  in  $w.\text{GETSTRUCTURALNODES}()$  do
5:       if ! $\text{erreichbareKnoten}.\text{HASKEY}(x)$  then
6:          $\text{erreichbareKnoten}[x] \leftarrow 0$ 
7:          $\text{erreichbareKnoten}[x] \leftarrow \text{erreichbareKnoten}[x] + 1$ 
8:    $T \leftarrow \text{MAX}(\text{erreichbareKnoten}, q)$ 
9:
10:   $s \leftarrow 0$ 
11:  for each Knoten  $x \in T$  do
12:     $s \leftarrow s + \text{erreichbareKnoten}[x]$ 
13:   $\text{relativeHaeufigkeit} \leftarrow$  leeres assoziatives Array
14:  for each Knoten  $x \in T$  do
15:     $\text{relativeHaeufigkeit}[x] \leftarrow \frac{\text{erreichbareKnoten}[x]}{s}$ 
16:
17:   $\text{random} \leftarrow \text{RANDOM}(0, 1)$ 
18:   $r \leftarrow 0.0$ 
19:   $i \leftarrow 0$ 
20:  while  $s < \text{random}$  do
21:     $r \leftarrow r + \text{relativeHaeufigkeit}[i]$ 
22:     $i \leftarrow i + 1$ 
23:   $v \leftarrow T[i - 1]$ 
24:  return  $v$ 

```

D. Vokabularbestimmung

Da die Größe des Vokabulars die Datenmenge signifikant beeinflusst, liegt es in unserem Interesse so wenig Wörter wie möglich ins Vokabular aufzunehmen. Insbesondere sind Wörter nicht von Interesse, die in fast allen Texten vorkommen, wie im Deutschen z. B. „und“, „mit“ und die Pronomen. Es ist

wünschenswert Wörter zu wählen, die die Texte möglichst stark voneinander unterscheiden. Der DYCOS-Algorithmus wählt die Top- m dieser Wörter als Vokabular, wobei $m \in \mathbb{N}$ eine festzulegende Konstante ist. In [AL11, S. 365] wird der Einfluss von $m \in \{5, 10, 15, 20\}$ auf die Klassifikationsgüte untersucht und festgestellt, dass die Klassifikationsgüte mit größerem m sinkt, sie also für $m = 5$ für den DBLP-Datensatz am höchsten ist. Für den CORA-Datensatz wurde mit $m \in \{3, 4, 5, 6\}$ getestet und kein signifikanter Unterschied festgestellt.

Nun kann man manuell eine Liste von zu beachtenden Wörtern erstellen oder mit Hilfe des Gini-Koeffizienten automatisch ein Vokabular erstellen. Der Gini-Koeffizient ist ein statistisches Maß, das die Ungleichverteilung bewertet. Er ist immer im Intervall $[0, 1]$, wobei 0 einer Gleichverteilung entspricht und 1 der größtmöglichen Ungleichverteilung.

Sei nun $n_i(w)$ die Häufigkeit des Wortes w in allen Texten mit der i -ten Knotenbeschriftung.

(1)

$$p_i(w) := \frac{n_i(w)}{\sum_{j=1}^{|\mathcal{L}_t|} n_j(w)} \quad (\text{Relative Häufigkeit des Wortes } w)$$

(2)

$$G(w) := \sum_{j=1}^{|\mathcal{L}_t|} p_j(w)^2 \quad (\text{Gini-Koeffizient von } w)$$

In diesem Fall ist $G(w) = 0$ nicht möglich, da zur Vokabularbestimmung nur Wörter betrachtet werden, die auch vorkommen.

Ein Vorschlag, wie die Vokabularbestimmung implementiert werden kann, ist als Pseudocode mit Algorithmus 4 gegeben. In Zeile 6 wird eine Teilmenge $S_t \subseteq V_{L,t}$ zum Generieren des Vokabulars gewählt. In Zeile 8 wird ein Array $cLabelWords$ erstellt, das $(|\mathcal{L}_t| + 1)$ Felder hat. Die Elemente dieser Felder sind jeweils assoziative Arrays, deren Schlüssel Wörter und deren Werte natürliche Zahlen sind. Die ersten $|\mathcal{L}_t|$ Elemente von $cLabelWords$ dienen dem Zählen der Häufigkeit der Wörter von Texten aus S_t , wobei für jede Beschriftung die Häufigkeit einzeln gezählt wird. Das letzte Element aus $cLabelWords$ zählt die Summe der Wörter. Diese Datenstruktur wird in Zeile 9 bis 15 gefüllt.

In Zeile 18 bis 20 wird die relative Häufigkeit der Wörter bzgl. der Beschriftungen bestimmt. Daraus wird in Zeile 21 bis 23 der Gini-Koeffizient berechnet. Schließlich werden in Zeile 24 bis 25 die Top- q Wörter mit den höchsten Gini-Koeffizienten zurückgegeben.

Die Menge S_t kann aus der Menge aller Dokumente, deren Knoten beschriftet sind, mithilfe des in [Vit85] vorgestellten Algorithmus bestimmt werden.

IV. ANALYSE DES DYCOS-ALGORITHMUS

Für den DYCOS-Algorithmus wurde in [AL11] bewiesen, dass sich nach Ausführung von DYCOS für einen unbeschrifteten Knoten mit einer Wahrscheinlichkeit von höchstens $(|\mathcal{L}_t| - 1) \cdot e^{-l \cdot b^2 / 2}$ eine Knotenbeschriftung ergibt, deren relative

Algorithmus 4 Vokabularbestimmung

Input:

- 1: $V_{L,t}$ (beschriftete Knoten),
- 2: \mathcal{L}_t (Menge der Beschriftungen),
- 3: $f : V_{L,t} \rightarrow \mathcal{L}_t$ (Beschriftungsfunktion),
- 4: m (Gewünschte Vokabulargröße)

Output: \mathcal{M}_t (Vokabular)

```

5:
6:  $S_t \leftarrow \text{SAMPLE}(V_{L,t})$   $\triangleright$  Wähle  $S_t \subseteq V_{L,t}$  aus
7:  $\mathcal{M}_t \leftarrow \emptyset$   $\triangleright$  Menge aller Wörter
8:  $cLabelWords \leftarrow$  Array aus  $(|\mathcal{L}_t| + 1)$  assoziativen Arrays
9: for each  $v \in S_t$  do
10:    $i \leftarrow \text{GETLABEL}(v)$ 
11:    $\triangleright w$  ist das Wort,  $c$  ist die Häufigkeit
12:   for each  $(w, c) \in \text{GETTEXTASMULTISET}(v)$  do
13:      $cLabelWords[i][w] \leftarrow cLabelWords[i][w] + c$ 
14:      $cLabelWords[|\mathcal{L}_t|][w] \leftarrow cLabelWords[i][|\mathcal{L}_t|] +$ 
15:        $c$ 
16:    $\mathcal{M}_t = \mathcal{M}_t \cup \{w\}$ 
17: for each Wort  $w \in \mathcal{M}_t$  do
18:    $p \leftarrow$  Array aus  $|\mathcal{L}_t|$  Zahlen in  $[0, 1]$ 
19:   for each Label  $i \in \mathcal{L}_t$  do
20:      $p[i] \leftarrow \frac{cLabelWords[i][w]}{cLabelWords[i][|\mathcal{L}_t|]}$ 
21:    $w.\text{gini} \leftarrow 0$ 
22:   for each  $i \in 1, \dots, |\mathcal{L}_t|$  do
23:      $w.\text{gini} \leftarrow w.\text{gini} + p[i]^2$ 
24:  $\mathcal{M}_t \leftarrow \text{SORTDESCENDINGBYGINI}(\mathcal{M}_t)$ 
25: return  $\text{TOP}(\mathcal{M}_t, m)$ 

```

Häufigkeit weniger als b der häufigsten Beschriftung ist. Dabei ist $|\mathcal{L}_t|$ die Anzahl der Beschriftungen und l die Länge der Random-Walks.

Außerdem wurde experimentell anhand des DBLP-Datensatzes² und des CORA-Datensatzes³ gezeigt (vgl. Tabelle I), dass die Klassifikationsgüte nicht wesentlich von der Anzahl der Wörter mit höchstem Gini-Koeffizient m abhängt. Des Weiteren betrug die Ausführungszeit auf einem Kern eines Intel Xeon 2.5 GHz Servers mit 32 GB RAM für den DBLP-Datensatz unter 25 s, für den CORA-Datensatz sogar unter 5 s. Dabei wurde eine für CORA eine Klassifikationsgüte von 82 %–84 % und auf den DBLP-Daten von 61 %–66 % erreicht.

Name	Knoten	davon beschriftet	Kanten	Beschriftungen
CORA	19 396	14 814	75 021	5
DBLP	806 635	18 999	4 414 135	5

Tabelle I: Datensätze, die für die experimentelle Analyse benutzt wurden

Obwohl es sich nicht sagen lässt, wie genau die Ergebnisse aus [AL11] zustande gekommen sind, eignet sich das Kreuzvalidierungsverfahren zur Bestimmung der Klassifikationsgüte wie es in [Lav06], [Sto74] vorgestellt wird:

²<http://dblp.uni-trier.de/>

³<http://people.cs.umass.edu/mccallum/data/cora-classify.tar.gz>

- 1) Betrachte nur $V_{L,T}$.
- 2) Unterteile $V_{L,T}$ zufällig in k disjunkte Mengen M_1, \dots, M_k .
- 3) Teste die Klassifikationsgüte, wenn die Knotenbeschriftungen aller Knoten in M_i für DYCOS verborgen werden für $i = 1, \dots, k$.
- 4) Bilde den Durchschnitt der Klassifikationsgüten aus Punkt 3.

Es wird $k = 10$ vorgeschlagen.

V. PROBLEME DES DYCOS-ALGORITHMUS

Bei der Anwendung des in [AL11] vorgestellten Algorithmus auf reale Datensätze könnten zwei Probleme auftreten, die im Folgenden erläutert werden. Außerdem werden Verbesserungen vorgeschlagen, die es allerdings noch zu untersuchen gilt.

A. Anzahl der Knotenbeschriftungen

So, wie der DYCOS-Algorithmus vorgestellt wurde, können nur Graphen bearbeitet werden, deren Knoten jeweils höchstens eine Beschriftung haben. In vielen Fällen, wie z. B. Wikipedia mit Kategorien als Knotenbeschriftungen haben Knoten jedoch viele Beschriftungen.

Auf einen ersten Blick ist diese Schwäche einfach zu beheben, indem man beim zählen der Knotenbeschriftungen für jeden Knoten jedes Beschriftung zählt. Dann wäre noch die Frage zu klären, mit wie vielen Beschriftungen der betrachtete Knoten beschriftet werden soll.

Jedoch ist z. B. bei Wikipedia-Artikeln auf den Knoten eine Hierarchie definiert. So ist die Kategorie „Klassifikationsverfahren“ eine Unterkategorie von „Klassifikation“. Bei dem Kategorisieren von Artikeln sind möglichst spezifische Kategorien vorzuziehen, also kann man nicht einfach bei dem Auftreten der Kategorie „Klassifikationsverfahren“ sowohl für diese Kategorie als auch für die Kategorie „Klassifikation“ zählen.

B. Überanpassung und Reklassifizierung

Aggarwal und Li beschreiben in [AL11] nicht, auf welche Knoten der Klassifizierungsalgorithmus angewendet werden soll. Jedoch ist die Reihenfolge der Klassifizierung relevant. Dazu folgendes Minimalbeispiel:

Gegeben sei ein dynamischer Graph ohne textuelle Inhalte. Zum Zeitpunkt $t = 1$ habe dieser Graph genau einen Knoten v_1 und v_1 sei mit dem A beschriftet. Zum Zeitpunkt $t = 2$ komme ein nicht beschrifteter Knoten v_2 sowie die Kante (v_2, v_1) hinzu.

Nun wird der DYCOS-Algorithmus auf diesen Knoten angewendet und v_2 mit A beschriftet.

Zum Zeitpunkt $t = 3$ komme ein Knoten v_3 , der mit B

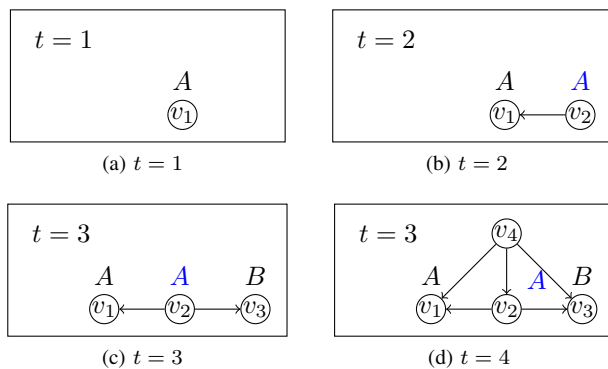


Abbildung 2: Minimalbeispiel für den Einfluss früherer DYCOS-Anwendungen

beschriftet ist, und die Kante (v_2, v_3) hinzu. Abbildung 2 visualisiert diesen Vorgang.

Würde man nun den DYCOS-Algorithmus erst jetzt, also anstelle von Zeitpunkt $t = 2$ zum Zeitpunkt $t = 3$ auf den Knoten v_2 anwenden, so würde eine 50 %-Wahrscheinlichkeit bestehen, dass dieser mit B beschriftet wird. Aber in diesem Beispiel wurde der Knoten schon zum Zeitpunkt $t = 2$ beschriftet. Obwohl es in diesem kleinen Beispiel noch keine Rolle spielt, wird das Problem klar, wenn man weitere Knoten einfügt:

Wird zum Zeitpunkt $t = 4$ ein unbeschrifteter Knoten v_4 und die Kanten (v_1, v_4) , (v_2, v_4) , (v_3, v_4) hinzugefügt, so ist die Wahrscheinlichkeit, dass v_4 mit A beschriftet wird bei $\frac{2}{3}$. Werden die unbeschrifteten Knoten jedoch erst jetzt und alle gemeinsam beschriftet, so ist die Wahrscheinlichkeit für A als Beschriftung bei nur 50%. Bei dem DYCOS-Algorithmus findet also eine Überanpassung an vergangene Beschriftungen statt.

Das Reklassifizieren von Knoten könnte eine mögliche Lösung für dieses Problem sein. Knoten, die durch den DYCOS-Algorithmus beschriftet wurden könnten eine Lebenszeit bekommen (TTL, Time to Live). Ist diese abgelaufen, wird der DYCOS-Algorithmus erneut auf den Knoten angewendet.

VI. AUSBLICK

Den DYCOS-Algorithmus kann in einigen Aspekten erweitert werden. So könnte man vor der Auswahl des Vokabulars jedes Wort auf den Wortstamm zurückführen. Dafür könnte zum Beispiel der in [Por97] vorgestellte Porter-Stemming-Algorithmus verwendet werden. Durch diese Maßnahme wird das Vokabular kleiner gehalten wodurch mehr Artikel mit einander durch Vokabular verbunden werden können. Außerdem könnte so der Gini-Koeffizient ein besseres Maß für die Gleichheit von Texten werden.

Eine weitere Verbesserungsmöglichkeit besteht in der Textanalyse. Momentan ist diese noch sehr einfach gestrickt und ignoriert die Reihenfolge von Wörtern beziehungsweise Wertungen davon. So könnte man den DYCOS-Algorithmus in

einem sozialem Netzwerk verwenden wollen, in dem politische Parteiaffinität von einigen Mitgliedern angegeben wird um die Parteiaffinität der restlichen Mitglieder zu bestimmen. In diesem Fall macht es jedoch einen wichtigen Unterschied, ob jemand über eine Partei gutes oder schlechtes schreibt.

Eine einfache Erweiterung des DYCOS-Algorithmus wäre der Umgang mit mehreren Beschriftungen.

DYCOS beschränkt sich bei inhaltlichen Zweifachsprüngen auf die Top- q -Wortknoten, also die q ähnlichsten Knoten gemessen mit der Aggregatanalyse, allerdings wurde bisher noch nicht untersucht, wie der Einfluss von $q \in \mathbb{N}$ auf die Klassifikationsgüte ist.

LITERATUR

- [AL11] C. C. Aggarwal and N. Li, “On node classification in dynamic content-based networks,” in *SDM*, 2011, pp. 355–366.
- [BCM11] S. Bhagat, G. Cormode, and S. Muthukrishnan. (2011) Node classification in social networks.
- [JCSZ10] C. Jiang, F. Coenen, R. Sanderson, and M. Zito, “Text classification using graph mining-based feature extraction,” *Knowledge-Based Systems*, vol. 23, no. 4, pp. 302 – 308, 2010, artificial Intelligence 2009 AI-2009 The 29th {SGAI} International Conference on Artificial Intelligence. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S095070510900152X>
- [Joa98] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” 1998.
- [Ko12] Y. Ko, “A study of term weighting schemes using class information for text classification,” in *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR ’12. New York, NY, USA: ACM, 2012, pp. 1029–1030. [Online]. Available: <http://doi.acm.org/10.1145/2348283.2348453>
- [Lav06] N. Lavesson, “Evaluation and analysis of supervised learning algorithms and classifiers,” Diploma Thesis, Blekinge Institute of Technology, Sweden, Dec. 2006.
- [NMTM99] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, “Text classification from labeled and unlabeled documents using em,” in *Machine Learning*, 1999, pp. 103–134.
- [Por97] M. F. Porter, “Readings in information retrieval,” K. Sparck Jones and P. Willett, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, ch. An Algorithm for Suffix Stripping, pp. 313–316. [Online]. Available: <http://dl.acm.org/citation.cfm?id=275537.275705>
- [SJ01] M. Szummer and T. Jaakkola, “Partially labeled classification with markov random walks,” in *Advances in Neural Information Processing Systems 14*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds., 2001, pp. 945–952. [Online]. Available: http://media.nips.cc/nipsbooks/nipspapers/paper_files/nips14/AA36.pdf
- [Sto74] M. Stone, “Cross-Validatory Choice and Assessment of Statistical Predictions,” *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 36, no. 2, pp. 111–147, 1974. [Online]. Available: <http://dx.doi.org/10.2307/2984809>
- [Vit85] J. S. Vitter, “Random sampling with a reservoir,” *ACM Trans. Math. Softw.*, vol. 11, no. 1, pp. 37–57, 1985. [Online]. Available: <http://doi.acm.org/10.1145/3147.3165>
- [ZG02] X. Zhu and Z. Ghahramani, “Learning from labeled and unlabeled data with label propagation,” Carnegie Mellon University, Tech. Rep., 2002.